AFRL-RI-RS-TR-2010-190

WIRELESS COMPUTING ARCHITECTURE II

Harvard University

November 2010

FINAL TECHNICAL REPORT

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■**UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

REPORT DOCUMENTATION PAGE	*Form Approved* **OMB No. 0704-0188**

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Service, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)* NOVEMBER 2010	**2. REPORT TYPE** Final Technical Report	**3. DATES COVERED** *(From - To)* April 2009 - April 2010

4. TITLE AND SUBTITLE WIRELESS COMPUTING ARCHITECTURE II	**5a. CONTRACT NUMBER** FA8750-09-2-0180
	5b. GRANT NUMBER N/A
	5c. PROGRAM ELEMENT NUMBER WCNA
6. AUTHOR(S) H.T. Kung	**5d. PROJECT NUMBER** WC
	5e. TASK NUMBER H2
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Harvard University, School of Engineering and Applied Sciences, 33 Oxford Street, Cambridge MA 02138	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RITB, 525 Brooks Road, Rome NY 13441-4505	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RITB
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-RI-RS-TR-2010-190

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. PA# 88 ABW-2010-5729
Date Cleared: 28 October 2010.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
In the next decade we will need large-scale and sophisticated information processing, such as near real-time signal analysis, machine learning and query processing, at the tactical edge that integrate sensor and intelligence collection. This calls for on-demand, instantly deployable computing resources to support on-going local operations in the field. Wireless computing and networking technologies that can cope with the relatively modest bandwidths of wireless links while being able to take advantage of their broadcast nature will need to be developed. We have obtained new results capable of addressing aspects of these challenges. First, our methodology of finite-state modeling can provide wireless link models for unmanned aerial vehicles (UAVs) in emulating UAV-based airborne computing systems. Second, our measurement results suggest that UAVs can exploit multi-receiver diversity gains to enhance communications. Third, our speculative pre-fetching approach hides I/O latency for situations where I/O operations with distant nodes can be a performance bottleneck, as in the case when a UAV interacts with other UAVs or ground nodes.

15. SUBJECT TERMS
Computation, Wireless Networks, Unmanned Aerial Vehicle, UAV, Airborne Networking, Receiver Diversity, Transmission Control Protocol, TCP, Cloud Computing, Map Reduce

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** WEN LEE
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	23	**19b. TELEPHONE NUMBER** (*Include area code)* N/A

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 ABSTRACT

For this project, we conducted research in theory and systems on wireless computing architectures. The work included establishment of theoretical models, development of system testbeds based on wireless local area networks, as well as system experimentation in the lab and in the field. It has led to a number of results and findings, including:

- A methodology for building finite-state models capable of predicting the performance of Transmission Control Protocol (TCP) over a varying wireless channel between an unmanned aerial vehicle (UAV) and ground nodes;
- Field measurements demonstrating that a UAV can use multiple receivers to realize substantial diversity gains that enhance the receipt of packets from ground nodes;
- A pipeline-based scheduling strategy, called *speculative pipelining*, which uses speculative prefetching and speculative computing to minimize execution delays caused by fluctuations in available computing and network capacities in shared compute clouds.

We have documented these results in several papers. Some of them will be published in the near future:

- "A Location-dependent Runs-and-gaps Model for Predicting TCP performance over a UAV Wireless Channel," Military Communications Conference (MILCOM 2010), November 2010.
- "Measuring Diversity on a Low-Altitude UAV in a Ground-to-Air Wireless 802.11 Mesh Network," IEEE Globecom 2010 Workshop on Wireless Networking for Unmanned Aerial Vehicles (Wi-UAV 2010), December 2010.
- "Speculative Pipelining for Compute Cloud Programming," Military Communications Conference (MILCOM 2010), November 2010.

2.0 INTRODUCTION

Our work has the goal of providing on-demand computing resources over wireless computing and communication infrastructures. We aim to develop practical system architectures for cloud computing that target the needs and usage modes of certain functions of the Air Force. Specifically, we hope to develop airborne compute clouds based on UAVs or other low-flying assets. In contrast to traditional approaches that rely on remote data centers that may become disconnected from users in the field, UAV-based compute clouds are capable of reliably delivering services on-demand at tactical edges.

These newly envisioned capabilities could be significant for many military scenarios. Consider, for example, decision systems at the tactical edge that integrate sensor and intelligence collection. These systems can address major challenges in intelligence, surveillance, and reconnaissance (ISR) applications, such as close-proximity sensing, on-demand multi-modal sensing and, generally, layered sensing [[1]]. Decisions in edge scenarios must often be made rapidly, even under complications arising from unpredictable application demands and large data sets. For example, application demands may change in response to sudden weather shifts or “surprise” events detected by surveillance and reconnaissance apparatus, and computing nodes on the edge must operate on large data sets that are often characterized by insufficient metadata and schema. These complications mean that new decision systems must not only provide better sensing capabilities, but also support large-scale and sophisticated information processing, such as near real-time machine learning of ontologies. These new capabilities are beyond the present technologies designed for traditional mission planning and forensic analysis, and require on-demand, instantly deployable computing resources to support on-going local operations in the field.

To realize this vision, providing proper computing and communications systems over wireless networks will be essential. Computing and networking technologies that can cope with the relatively modest bandwidths of wireless communications channels while being able to take advantage of their broadcast nature will need to be developed.

In this project, we have obtained new results capable of addressing aspects of these challenges. First, our methodology of finite-state modeling can provide wireless link models for unmanned aerial vehicles (UAVs) when simulating or emulating UAV-based airborne computing systems.

Second, our results exploiting multi-receiver diversity gains suggest an effective method of enhancing UAV communications over wireless links in support of wireless computational architectures. Third, the speculative prefetching approach we have developed hides I/O latency for situations where I/O operations with distant nodes can be a performance bottleneck, as in the case when a UAV interacts with other UAVs or ground nodes.

In the rest of this section, we give a brief overview of our results in these three areas. As we have an interest in employing commercial-off-the shelf (COTS) components [[2]], we use the IEEE 802.11 (aka Wi-Fi) medium access control and physical layer [[3]] throughout.

2.1 Finite-State Models for UAV Links

We have developed a methodology of constructing finite-state models to predict the performance of TCP over a varying wireless channel between a UAV and ground nodes. As a UAV traverses its flight path, the wireless channel may experience periods of significant packet loss, successful packet delivery, and intermittent reception. By capturing packet run-length and gap-length statistics at various locations on the flight path, this location-dependent model can predict TCP throughput in spite of dynamically changing channel characteristics. We train the model by using packet traces from flight tests in the field and validate it by comparing TCP throughput distributions for model-generated traces against those for actual traces randomly sampled from field data. Our modeling methodology is general and can be applied to any UAV flight path.

2.2 Multi-Receiver Gains for UAV Communications

We consider the problem of mitigating a highly varying wireless channel between a transmitting ground node and receivers on a small, low-altitude UAV in a wireless mesh network based on the 802.11 standard for wireless local area networks. One approach is to use multiple transmitter and receiver nodes that exploit the channel's spatial/temporal diversity and that cooperate to improve overall packet reception. We have obtained a series of measurement results from a real-world testbed that characterize the resulting wireless channel. We show that the correlation between receiver nodes on the airplane is relatively insignificant at small time scales. That is, at any given time some of the receivers can receive packets well, while others may not. This means that receiver diversity can be exploited. Our measurements suggest that using several receiver nodes simultaneously can boost packet delivery rates substantially. Lastly, we show that similar results apply to transmitter selection diversity as well.

2.3 Speculative Pipelining in Compute Clouds

As cloud computing services have come online, new programming models tailored to the compute cloud have evolved in tandem. Applications using MapReduce [[4]], a programming model for parallel data processing on large-scale commodity computing clusters, have formed a significant category on compute clouds like Amazon's Elastic Compute Cloud (EC2), where data center resources can be rented by the hour. For the defense and intelligence communities, compute cloud programming models like MapReduce may be a cost-effective way for processing the extremely high volume of sensor data streaming from the tactical edge [[5]].

MapReduce job execution typically occurs in sequential phases of parallel steps. These phases can experience unpredictable delays when available computing and network capacities fluctuate or when there are large disparities in inter-node communication delays, as can occur on shared compute clouds. We propose a pipeline-based scheduling strategy, called speculative pipelining, which uses speculative prefetching and speculative computing to minimize execution delays in subsequent stages due to varying resource availability. Our proposed method can mask the time required to perform speculative operations by overlapping with other ongoing operations. We introduce the notion of "open-option" prefetching, which, via coding techniques, allows speculative prefetching to begin even before knowing exactly which input will be needed. On a compute cloud testbed, we apply speculative pipelining to the Hadoop sorting benchmark and show that sorting time is shortened significantly.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

In this section, we describe assumptions and methodologies we used in the three areas of work.

3.1 Finite-State Models for UAV Links

We consider using TCP to transport data from ground nodes to a low-flying, fixed-wing UAV over a wireless channel. We are interested in constructing a finite-state model for predicting TCP throughput over the wireless channel that will enable repeatable, realistic emulation to support application and system development. This model must address the large variations in the wireless channel that the UAV may experience over the course of its flight path, and allow the emulator to use the real TCP stack for congestion control [[6]].

One of the challenges in capturing channel conditions is to account for physical and environmental effects that may cause packet loss/reception events to be correlated over time [[7]]; in modeling, this effect is often referred to as channel memory. In periods where the wireless channel sees significant packet loss, TCP throughput is largely determined by patterns of consecutive loss (gaps) and consecutive reception (runs) events that are dictated by channel memory, rather than simply the average packet loss rate over a period.

The large variations in loss behavior that we observe with respect to the position of the UAV suggest a location-dependent model is necessary, where distinct submodels are used to characterize different legs of the flight path. Practically, we must be able to train each submodel with a relatively small amount of data, as we are subject to the velocity of and area covered by the plane.

Our modeling approach can meet these objectives. In the past, other modeling efforts employed multi-state Markov chains, or simply drew run and gap lengths from a distribution fitted to measured data. However, we show that these existing methods can be further improved by grouping contiguous locations along the flight path with similar run and gap statistics (referred to as "legs"), and then clustering within legs based on run/gap distributions. We train a different submodel for clusters within each leg, which can later be combined to produce accurate trace-based TCP throughput predictions despite limited training data availability. Figure 1 depicts this procedure.

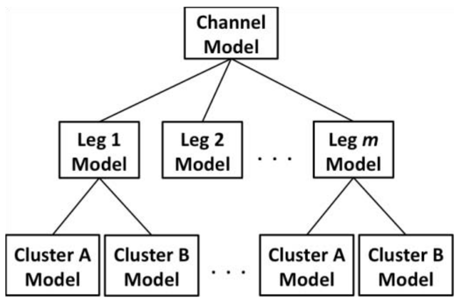


Figure 1. The decomposition of our wireless channel model into component submodels.

After the model is built, we use it to generate synthetic packet traces based on the control flow shown in Figure 2.

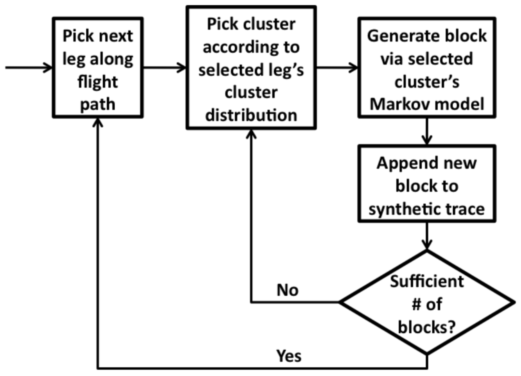

Pick next leg along flight path
Pick cluster according to selected leg's cluster distribution
Generate block via selected cluster's Markov model
Append new block to synthetic trace
Sufficient # of blocks?
No
Yes

Figure 2. The process our model employs to generate a synthetic packet trace along the modeled flight path.

3.2 Multi-Receiver Gains for UAV Communications

To study the receiver diversity gain attainable by multiple receiver nodes on a UAV, we collected ground-to-air packet traces by recording individual packet transmissions and receptions originating from a fixed ground transmitter node to four UAV-mounted receiver nodes, respectively. We have applied a similar approach in the air-to-ground direction, and will report those results in the future.

We first describe our experimental setup for collecting these traces. The UAV we used had a 110” wingspan, a radio-transparent fuselage, and aluminum landing gear and wing struts. Its electric engine was powered by batteries that permit a maximum flight time of 20 minutes. The UAV also had an autopilot that follows a flight path defined by waypoints. With low wind, the autopilot achieves high positional accuracy across laps—e.g., we have found positional deviation with respect to a fixed waypoint to be no more than 13m when the plane was level, and no more than 25m when the plane was banking through a turn. This permitted repeatable data collection over multiple laps within a flight and over multiple flights, as illustrated by the flight traces in Figure 3. During a flight, we logged altitude, airspeed, Global Positioning System (GPS) latitude/longitude, and GPS time every 250ms.

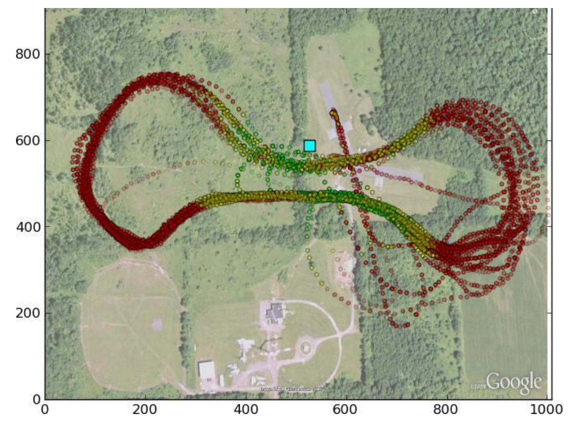


Figure 3. The UAV flight path used in our experiments.

At an altitude of 75m, our UAV followed a cyclical, dumbbell-shaped flight path that passed beyond the radio range at the extremes (3). Each lap ran clockwise and, at airspeed of 20m/s, lasted about 90s. Cruising time for each flight was about 15 minutes, meaning the UAV covered ten laps per flight. We collected packet traces over four flights in total. These flights were conducted at a property surrounded by woods and farmland, free of 802.11 radio traffic.

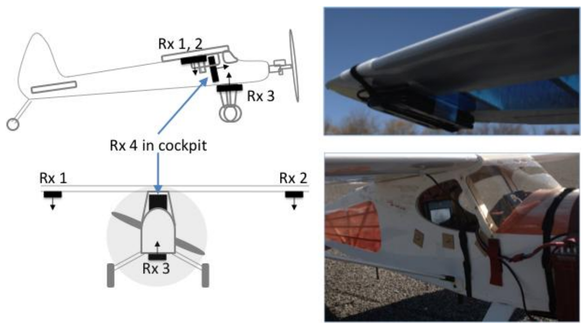


Figure 4. The mounting positions of the MIDs on the UAV; arrows show the direction the MID screen faces.

Our wireless mesh network nodes were Mobile Internet Devices (MIDs) that feature an 800MHz x86-compatible Intel Atom processor, a Marvell SD8686 802.11b/g internal Secure Digital Input Output (SDIO) radio, an internal omnidirectional antenna and Ubuntu Linux 8.04. The antenna orientation was determined by the direction at which the MID screen faces. One transmitter MID was placed amid tall grass and brush, elevated 20cm from the ground, with its screen facing skyward. Its location is shown as a square in 3. Four receiver MID nodes (Rx1 to Rx4) were mounted on the UAV as shown in Figure 4. Rx1 and Rx2 were mounted with screens facing the ground, one on the underside of each wing tip. Rx3, with its screen facing skyward, was mounted between the landing gear and Rx4, with its screen facing the bow, was mounted inside the cockpit. The direction of mounting is significant since measurements we took in an anechoic chamber show that the MID's antenna pattern is asymmetric. Thus, our mounting configuration provides diversity in antenna patterns and offers different shadowing profiles from the landing gear, engine and batteries. We placed all these 802.11 nodes in ad-hoc mode using the 1 megabit-per-second modulation. To collect traces, our transmitter broadcast 1420-byte UDP packets, with unique sequence numbers, at 80 packets per second. The receiver nodes on the UAV ran a client that logged the sequence number and timestamp of each received packet.

3.3 Speculative Pipelining in Compute Clouds

In MapReduce, work is decomposed by the application developer into map and reduce tasks, which are executed in several sequential phases of parallel steps (Figure 5). While this is an elegantly simple way to perform processing of arbitrary big data, existing implementations of MapReduce are predicated on being hosted in private data centers (e.g., Google's), where a single administrative authority manages all resource needs and consumption of computing and network resources by unplanned competing applications can be minimized by fiat. As a result, some fundamental design choices and assumptions on the underlying resource pool in these implementations are a mismatch for shared compute clouds (e.g., Amazon EC2), where the resource availability can be highly unpredictable due to varying competing loads.

We seek to address these issues. To mitigate problems due to varying and unbalanced resource availability in computation and I/O, we speculatively execute future tasks using otherwise idle resources. First, during compute-intensive phases of a MapReduce job, we utilize an idle network to perform speculative prefetching of input data, overlapping computation with I/O. Thus, in future pipeline stages, a task's input will be readily available as a result of the prefetching. This is in contrast to the present MapReduce approach of reactive data fetching [[4]], which is limited in that the fetching occurs at arbitrary points in time, causing an input delay for the associated computation. This delay can be substantial when network resources are unavailable or severely limited, as in the case of UAV-based networks. Second, we wish to minimize the amount of cross-rack traffic that is on the critical-path of a MapReduce computation. During I/O-intensive phases of a MapReduce job, we perform speculative computing to increase the likelihood that map outputs will be rack-local to subsequent reducers, thus decreasing cross-rack network traffic. In other words, we trade additional computation, performed on otherwise idle CPUs, for reduced cross-rack traffic. This optimization is orthogonal to speculative prefetching, which hides I/O time rather than reduces I/O.

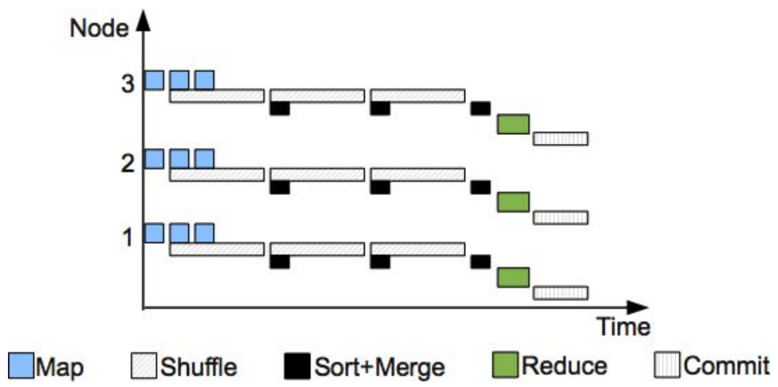


Figure 5. MapReduce job progress occurs over multiple nodes in sequential phases: map, shuffle, sort+merge, reduce, and commit.

4.0 RESULTS AND DISCUSSION

In this section, we present highlights of our results in the three areas.

4.1 Finite-State Models for UAV Links

In order to evaluate the performance of our models, we conduct trace-based emulation to estimate TCP throughput using both model-generated synthetic traces, and real traces from the UAV flights. Specifically, we instrument two Linux machines to redirect a TCP flow under test to a user-mode tunnel program which 1) exerts a rate limit equal to that in the trace collection, and 2) forwards or drops packets based on the nearest packet reception event in the trace [[8]] [[9]].

We performed the TCP emulation over a leg of the flight with good reception, i.e., few blocks with gaps spanning the entire block. We test finite-state models with orders n = 1, 2 and clusters k = 1, 2, as well as a validation trace, and present the TCP performance results in Table 1. We can see that the order n = 1 models underperform all order n = 2 models. For models of order n = 2, a k = 2 cluster model slightly outperforms the k = 1 cluster model. We observe that this improvement is small because the leg does not show strong clustered behavior, since it is chosen to have few long gaps in order to have sustained TCP throughput.

Trace Description	TCP Throughput (KB/s)	Validation Error
Validation trace	11.4	
Model $n = 1, k = 1$	13.7	20%
Model $n = 1, k = 2$	13.9	22%
Model $n = 2, k = 1$	10.4	9%
Model $n = 2, k = 2$	12.2	7%

Table 1. TCP performance observed on several synthetic traces and a validation trace. The validation error is computed with respect to the validation trace. Each value is an average of four runs; in the case of validation traces, each run used a different subset of the validation data.

4.2 Multi-Receiver Gains for UAV Communications

To determine the gain due to receiver diversity, we combine the packet receptions at the four receivers on the UAV into a combined trace, where a packet is marked as received if it was received by at least one individual receiver. Table 2 shows the diversity performance in terms of the overall delivery rates of combined traces for each flight, alongside the best single receiver's delivery rate. We can see that diversity brings about an additional gain of roughly 25%.

	Flight 1	Flight 2	Flight 3	Flight 4
Best single receiver	Rx2	Rx2	Rx2	Rx2
Best single delivery rate	0.44	0.37	0.38	0.40
Combined delivery rate	0.55	0.49	0.46	0.51
Percent improvement	25%	32%	21%	28%

Table 2. Combined packet delivery rates achieved by the four receivers during four separate flights and the resulting diversity gains.

4.3 Speculative Pipelining in Compute Clouds

We built a 46-node cluster of commodity machines powered by VIA VB8001 motherboards and VIA Nano 1.6GHz CPUs. Each board is connected to a switch via a 100Mbps Ethernet link. Lastly, each board is equipped with 1GB of RAM and 8GB of flash storage. Figure 6 shows photos of the cluster. We used either one or two switches to configure a one- or two-rack system, respectively. By using two switches we can create cross-rack bottleneck links typical in large data centers. Each switch was a Cisco Catalyst 3500 Series XL non-blocking 48-port switch, with all ports set to a fixed, 100 Mbps bit rate. Where we used two switches, the switches were bridged via a 100 Mbps Ethernet link. The nodes ran Hadoop v0.20.2. One node in the rack is designated the master and runs both the Hadoop and Hadoop Distributed File System (HDFS) master daemons, but does not execute any map or reduce tasks or store any HDFS data blocks. All other nodes run both Hadoop and HDFS slave daemons and are responsible for running computation tasks as well as storing HDFS data blocks. This arrangement is consistent with that of Amazon Elastic MapReduce clusters. In the experiments below, we use the existing MapReduce sort implementation found in the Hadoop code base, and previously used as a benchmark [[10]]. We apply speculative pipelining principles to improve its performance.

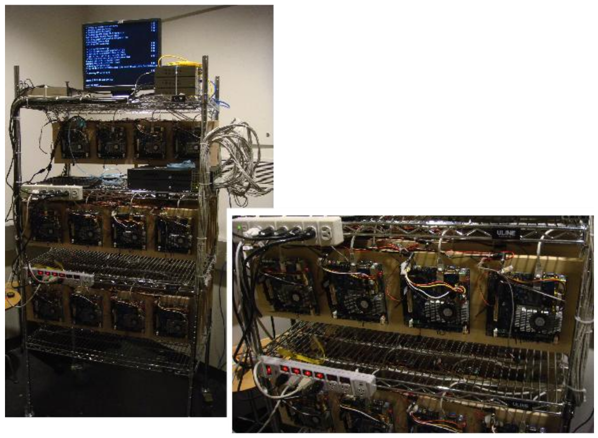


Figure 6. Hadoop cloud computing testbed of 46 nodes.

First, we demonstrate map phase behavior when no speculative prefetching is done. Figure 7a shows the map task completion time distribution in a control experiment where no exogenous CPU load was introduced on node *n1*. The spread of map completion times is fairly narrow (22s to 33s). In contrast, when we introduce exogenous CPU workload to *n1* and task *t1* is speculated to *nx*, *t1*'s completion time inflates to 213s as shown in Figure 7b. This is due to Hadoop's 60s straggler detection threshold and the time needed to fetch the input split from *n1* to *nx*. Note that the longest map task bounds the duration of the entire map phase, meaning this single straggling task impedes the entire job. Next, we emulate speculative prefetching by increasing the replication factor in HDFS such that any node chosen to speculatively re-execute *t1* will have a local copy of its input split. This is an indirect measurement method but accurately reflects our scenario, where the speculative prefetch time is completely masked by the map computation, and is thus equivalent to having a local replica of *t1*'s input split available at *nx* at the time of task speculation. Figure 7c shows the map task completion time distribution with speculative prefetching enabled. Note that *t1* now only takes 89s (60s straggler detection time + 29s map computation time) to finish, clearly demonstrating the advantage of speculative prefetching.

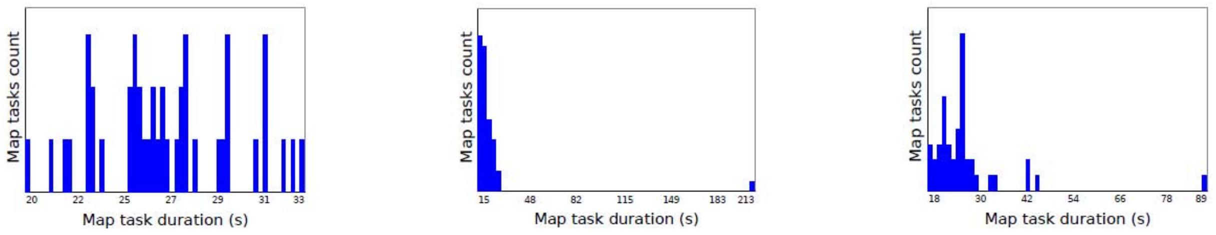
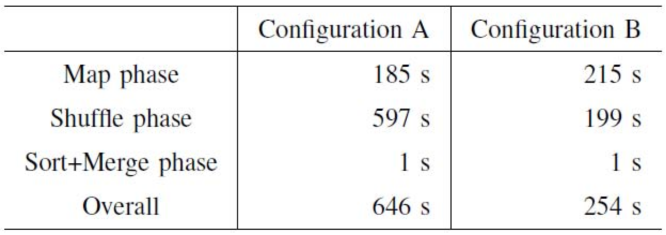


Figure 7. Map task duration distributions. The total number of map tasks is 44 in all three plots. (a) Baseline case, with no exogenous CPU load on node n1. (b) With exogenous CPU load on node n1 but without speculative prefetching. (c) With exogenous CPU load on n1 and with speculative prefetching, the overall map phase completion time is reduced from 213s to 89s.

As mentioned earlier, we can alleviate some cross-rack network load by performing extra computation to create rack-local replicas of the input data. For example, in the sorting application on a two-rack setup, mappers send their output to reducers in both racks. This involves cross-rack flows. However, suppose that the input data has two replicas per block, one on each rack. For such blocks, instead of running one mapper, we can run one mapper per replica, making the results available on both racks without transferring them across the inter-rack link.

We demonstrate the potential performance gain due to redundant mappers by running the unmodified Hadoop Sort application with parameters tuned to emulate a real implementation of speculative redundant computing as follows. First, we divide our testbed into two 22-node worker groups, each connected to a separate non-blocking switch. The switches are bridged with one 100Mbps link. An additional node, serving as the master, is also connected to one of the switches. We create input data such that each worker holds a 512MB slice, saved as four 128MB blocks. Then, we run Sort in two configurations: (A) using 44 workers, which results in a standard MapReduce sorting run on a 22Gb data set, and (B) using only one 22 worker group, sorting an 11Gb data set. We argue that the traffic pattern of the shuffle phase of Configuration B is equivalent to that of one rack of Configuration A, where every mapper is run redundantly, since the total amount of data transferred from mappers to reducers in one rack is the same, and there is no cross-rack traffic in either case. We show the running times of the two configurations in Table 3, broken down by MapReduce phase. Clearly, Configuration B gained significant speedup from reduced cross-rack traffic.



	Configuration A	Configuration B
Map phase	185 s	215 s
Shuffle phase	597 s	199 s
Sort+Merge phase	1 s	1 s
Overall	646 s	254 s

Table 3. Duration of map, shuffle, sort+merge, and reduce phases in Sort. Note that the overall duration is not the sum of individual phase durations, since the phases overlap.

5.0 CONCLUSIONS

Here we summarize our results and findings in the three areas of work.

5.1 Finite-State Models for UAV Links

By using a location-dependent runs-and-gaps model, we can predict TCP throughput over a varying ground-to-UAV wireless channel. This is achieved in spite of large channel variations resulting from dynamically changing conditions in communication distance, antenna angles, engine shadowing, etc., and TCP's sensitivity to distributions of packet loss (gaps) and delivery (runs) statistics. Two factors contribute to the success: (1) model training on clustered blocks of traces based on a relatively stable metric—mean gap-length, and (2) per leg modeling, leveraging a stable cluster distribution in a local geographic neighborhood. This is a general methodology for modeling TCP behavior and can be applied to any UAV flight path.

With our modeling methodology, we can construct link models for various UAV-to-ground links as well as UAV-to-UAV links. Based on these link models, we can build a system simulation/emulation system comprised of multiple UAV and ground nodes. For example, an ensemble of UAV nodes in the air and nodes on the ground can be viewed as a distributed MapReduce engine. Ground nodes can perform Map operations on data local to these nodes (e.g., data from nearby ground sensors). Via wireless links, UAVs can pick up the Map task results from ground nodes and execute Reduce tasks that consume these local results to achieve a global objective; such an arrangement is natural, since the UAV can have global access to the locally processed results. These simulation/emulation systems can assess the effectiveness of such a distributed airborne MapReduce system for real-world applications such as decision-making at tactical edges.

5.2 Multi-Receiver Gains for UAV Communications

Based on an experimental study of ground-to-air UAV links, we found that the links are intermediate in their qualities for a significant amount of time, owing to the UAV moving in and out of range frequently; this might be a property specific to fixed-wing craft, but is important nonetheless as those make up the most efficient class of fliers. As a result, network layer protocols will need to be adapted to cope with this situation, or to work on top of link layer mechanisms which use, e.g., coding or retransmissions to mask the loss [[11]].

At small time scales of up to 1000 packets, or about 10 seconds, we found that the packet losses seem to be almost uncorrelated, giving diversity performance very similar to that of synthetic traces generated using Bernoulli trials. This seems to indicate that the effect of fast fading is not significant in our environment in any direction, in the sense that it does not result in either better or worse diversity performance than that of memoryless losses.

The diversity gain we observed with four onboard receiver nodes consisted of an average 25% increase in delivery rate, and an additional 12% with three transmitter nodes using selection diversity. On the receiver side, a question arises whether this is sufficient to warrant the addition of extra receivers on a UAV; in the event that the multiple nodes are already present in the UAV application, using them for diversity reception would provide a clear and inexpensive benefit. Finally, on the transmitter side, using multiple transmitters introduces protocol complications related to feedback, synchronization, packet retransmissions, etc. However, as has been shown in previous work [[12]], many of these issues can be addressed using network coding techniques which reduce the need for frequent feedback.

5.3 Speculative Pipelining in Compute Clouds

We have presented an approach, called speculative pipelining, which can mitigate the impact of uncertain or insufficient resource availability on MapReduce job completion time. We have described two specific schemes: (1) speculative prefetching to reduce input delay when recovering from straggler tasks, and (2) speculative computing to reduce traffic traversing the cross-rack network bottleneck. We have shown the effectiveness of these schemes through a sorting benchmark run on a lab testbed. Furthermore, we have shown performance analyses and introduced novel ideas such as open-option speculative prefetching. Our speculative approach differs from current practices found in MapReduce implementations, where input fetching takes place only when input is needed. With speculative pipelining, we prefetch input to reduce future input delay, and perform redundant computation to reduce future network congestion. These speculative operations put otherwise unused CPU and network resources to good use and, in principle, can be extended to compute cloud programming models beyond MapReduce.

5.4 Concluding Remarks

We conclude with two overall remarks. First, modeling via experimentation as we have done in all three areas discussed above is a powerful methodology in addressing difficult issues in our wireless-related problem domain. For example, we have modeled UAV links and receiver diversity via packet traces obtained from actual flight tests. We have studied scheduling in cloud computing via modeling variations in available resources. Second, our work in systems links naturally to information-theoretical techniques and results. For example, we have used coding in “open-option” prefetching in scheduling for cloud computing. In addition, our results in receiver diversity and link modeling have led us to exploit coding-based systems. We plan to further explore these synergies in our future work.

6.0 REFERENCES

[1] M. Bryant, P. Johnson, B. M. Kent, M. Nowak and S. Rogers, "Layered Sensing: Its Definition, Attributes and Guiding Principles for AFRL Strategic Technology Development", Air Force Research Laboratory Report, May, 2008.

[2] D. Hague, H. T. Kung, and B. W. Suter, "Field experimentation of COTS-based UAV networking," in MILCOM, 2006.

[3] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, IEEE Std., 1999.

[4] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," in OSDI, 2004.

[5] D.Meiron, S.Cazares, P.Dimotakis, F.Dyson, D.Eardley, S. Keller-McNulty, D. Long, F. Perkins, W. Press, R. Schwitters, C. Stubbs, J. Tonry, and P. Weinberger, "Data analysis challenges," JASON, Tech. Rep. JSR- 08-142, Dec 2008.

[6] R. Caceres and L. Iftode, "Improving the performance of reliable transport protocols in mobile computing environments," IEEE Journal on Sel. Areas in Comm., vol. 13, no. 5, pp. 850–857, 1995.

[7] L. Ahumada, R. Feick, R. Valenzuela, and C. Morales, "Measurement and characterization of the temporal behavior of fixed wireless links," IEEE Trans. Vehic. Tech., vol. 54, no. 6, pp. 1913–1922, 2005.

[8] B. D. Noble, M. Satyanarayanan, G. T. Nguyen, and R. H. Katz, "Trace-based mobile network emulation," in SIGCOMM, 1997.

[9] M. Satyanarayanan and B. Noble, "The role of trace modulation in building mobile computing systems," in HotOS, 1997.

[10] O. O'Malley, "Terabyte sort on apache hadoop," http://sortbenchmark.org/YahooHadoop.pdf.

[11] H. T. Kung, C.-K. Lin, T.-H. Lin, S. J. Tarsa, and D. Vlah, "A location-dependent runs-and-gaps model for predicting TCP performance over a UAV wireless channel," in MILCOM, 2010 (to appear).

[12] H. T. Kung, C.-K. Lin, T.-H. Lin, S. J. Tarsa, and D. Vlah, "Flowcode: Multi-site data exchange over wireless ad-hoc networks using network coding," in MILCOM, 2009.

7.0 LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

DoD	Department of Defense
IEEE	Institute of Electrical and Electronics Engineers
GPS	Global Positioning System
HDFS	Hadoop Distributed File System
MAC	Medium Access Control
MID	Mobile Internet Device
RSSI	Received Signal Strength Indicator
SDIO	Secure Digital Input Output
TCP	Transmission Control Protocol
UAV	Unmanned Aerial Vehicle
Wi-Fi	Wireless Fidelity, a term for certain types of wireless local area networks that use specifications in the 802.11 standard